CS32420 ASSIGNMENT

Computer Graphics and Games

Room Modelling and Navigation

Jake McDonagh
jam93@aber.ac.uk

CS32420 Assignment – Room Modelling and Navigation

Jake McDonagh

Aberystwyth University, Wales
jam93@aber.ac.uk

Abstract. The following reported focuses on the production and creation of the CS32420 assignment in it's entirety. Within this report, the methods and techniques that I have used will be explained in detail. Throughout this report I will reference any material that I have used in an appropriate manner and how the original source code has been modified so that I could solve the more complicated problems that I had encountered during this assignment. Finally, I will conclude on the work that I had produced and will discuss about how I think the assignment went as well as the areas that I feel went well as well as the areas in which I feel I could have improved upon.

Keywords: Three.js, WebGL, Advanced Lighting, Texture Mapping, Complex Geometries, Meshes, Report, Shadow Mapping, Shadow Casting. Advanced Shadows.

1 Introduction

The following report is created in conjunction with the practical aspects of this assignment. This report will contain instructions on how to run the scene from a local machine to talking about the different aspects from the assignment which will include the modelling, textures and lighting as well as the advanced methods used such as shadow mapping, advanced lighting and complicated geometrical shapes.

1.1 Summary of Methods used within scene

Below is a summary of the methods that I have portrayed and used for this assignment. I will go into more detail later about each individual aspect later in the report.

- Basic Geometrical Shapes
- Advanced Geometrical Shapes
- Basic Texture Mapping
- Advanced Texture Mapping
- Advanced Photorealistic Lighting
- Modelling
- Basic Animation

2 Instructions on how to run and navigate scene

2.1 How to run the scene

To run the scene, unzip the downloaded folder and open the jam93CS32420.html file to begin. Due to the shadow mapping that is used within the source code, the file requires that WebGL is running. For most modern computers, this should already happen by default. Once the file has been opened, please wait a few seconds whilst the textures, lighting and shadows fully load for optimal performance.

The navigation is controlled by an OrbitControls.js file located within the ‘js’ file of the zip. Click and hold the left mouse button to look around the screen, whilst you can move around by clicking and holding the right mouse button. The mouse wheel can be used to zoom in and out of the scene as you see fit. There is no object collision within the scene, so the user will be able to effortlessly navigate the scene without too much hassle.

The .html file has been tested on both Google Chrome and Mozilla Firefox. While the file works on both internet browsers, it is highly recommended that you use Mozilla Firefox for marking purposes. This is due to the fact that in the case of Google Chrome, the textures used within the file don’t load as intended, whereas they load perfectly fine in Firefox Mozilla.

2.2 Summary of Zip File.

Within the zip file, there are 3 folders named ‘images’, ‘js’ and ‘objects’ respectively. Outside of those folders there are also three files called jam93CS32420.html, myCSS.css and this report, jam93CS32420.pdf. The myCSS files ensures that the .html file remains simple and attractive to the user as they open the file whereas the jam93CS32420.html is the practical aspect of this assignment.

The images folder contains all of the images used within the house for the purpose of texture mapping the objects to make them more aesthetically pleasing or to be used in order to make the props used look more believable. These image files will be appropriately referenced at an appropriate section of this report. Please see that section for more information on images used.

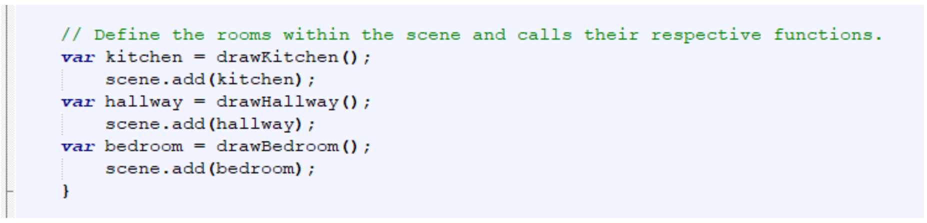
The js folder contains all of the three.js libraries used for this assignment. To reduce the overall file size of the zip and to ensure that the .html file is lightweight, I chose to only use the JavaScript files that I needed for this assignment. The JavaScript files are called into the jam93CS32420.html file via the script tags towards the very top of the file.

The objects folder contains a few object.json files that are loaded and rendered into the scene. The objects in this case (being a double-sided bed and a white chair) will be appropriate referenced within the Bedroom section of this report as the models are used within that room. I simply wanted to have multiple methods of creating objects within my assignment so for the purpose of trying out different methods, I used them within the scene.

3 Structure of JavaScript Code

The overall structure of the JavaScript code within the .html file consists of several functions that are called throughout. At the top level, there are two functions called initScene and renderScene. As you could derive from these functions, initScene initializes the entire scene at runtime as it ensures that everything has properly loaded up upon opening the file. The majority of the code that I have created and modified is found within this function. Likewise, renderScene ensures that the scene is being constantly updated as the user navigates the rooms. RenderScene applies to all aspects of animation from the movement and rotation of objects to the movement and rotation of the camera itself.

Within the initScene function, the renderer, scene and camera are all initialized and created using the global variables that can be found at the top of the JavaScript section of the file. Afterwards, the navigation is initialized and the scene is generated using the drawKitchen, drawHallway and drawBedroom functions.



```
// Define the rooms within the scene and calls their respective functions.
var kitchen = drawKitchen();
    scene.add(kitchen);
var hallway = drawHallway();
    scene.add(hallway);
var bedroom = drawBedroom();
    scene.add(bedroom);
}
```

Fig. 1. The methods within the JavaScript file that generates the rooms within the scene.

4 Kitchen

This section of the scene uses the following: cat.png, ceiling.jpg, dice0.png, dice1.png, dice2.png, dice3.png, dice4.png, dice5.png, kingdomDeathAI.png, kingdomDeathBoard.jpg, kingdomHL.jpg, kingdomDeathHL.png, tree.png, whitewall.jpg, wood.jpg.

4.1 The furniture

This room is the very first room that the user is going to see because said user initially spawns here. From the following picture below, this room makes great use of shadow mapping to ensure that there are high quality shadows where they should logically be. The room also makes great use of several different types of textures. The floors use a wooden texture [1] whilst both the walls and ceiling use a texture to simulate a painted white wall [2]. Both of the above textures have also been bump mapped in order to give the textures a more photorealistic, 3-dimensional look. There is a table within the center of the room that is created by using a BoxBufferGeometry object in conjunction with CylinderBufferGeometry objects to produce a table that is similar to an ordinary kitchen table. The table is also surrounded by 4 identical chairs that I had made using several different types of geometrical objects in order to produce a single complex object. The chair geometry can be found within the drawChair function. Both the table and the chairs use a brown colour with a MeshPhongMaterial to give the furniture a glossy look to them. Although I could have given the furniture a texture as well, I made the choice not to as I liked the glossy light effect that reflected off the table.

The other furniture in the room consists of a detailed drawer that was also made by combining multiple geometries and wireframes together to resemble a common household drawer. There are also three picture frames within the room that have pictures of a cat [3], a picture of my own copy of Kingdom Death: Monster which I will go into detail on in the next section as well as a picture of a tree at sunset [4]. The frames are created using thin BoxBufferGeometry objects that resemble a frame with a black mesh. The pictures are then placed closely next to object in question to give the illusion of a picture. The room also has a beige coloured lampshade attached to the ceiling of the room with a point light source just underneath it. All the objects within the room and the light source itself has its shadow mapping set up so that the room can render high quality shadows.

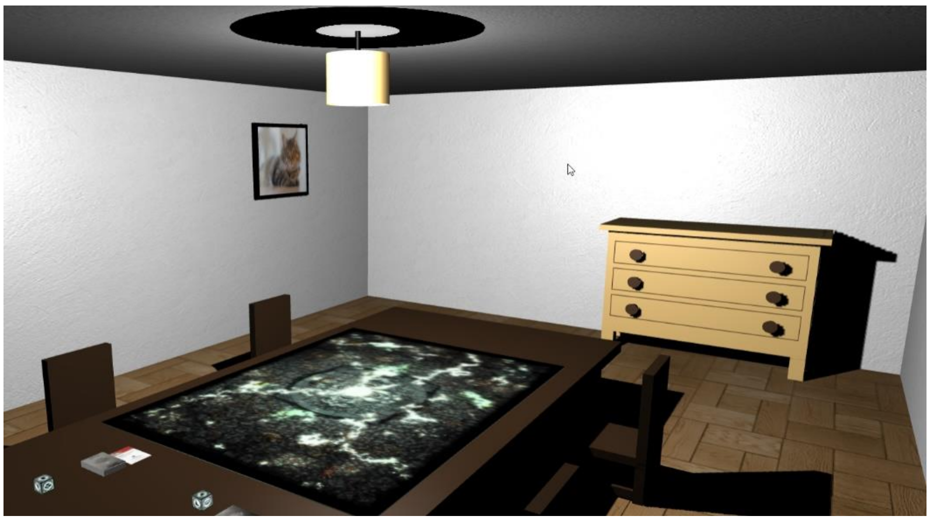
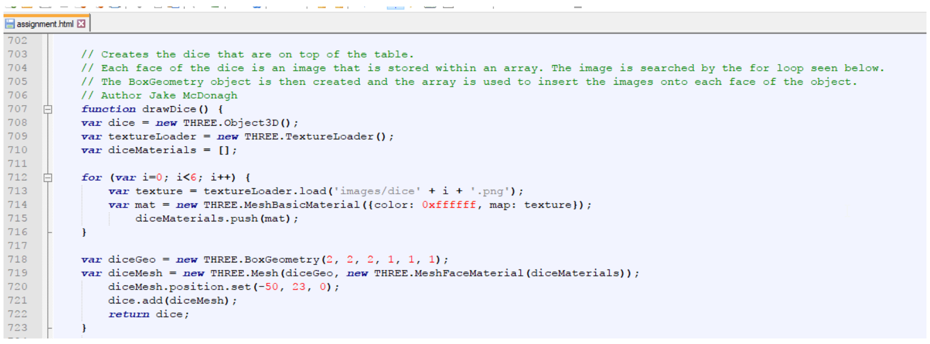


Fig. 2. An example view of the kitchen room.

4.2 The Table Objects

The top of the table consists of several different props that all have a relation to the tabletop game called Kingdom Death: Monster [5]. The props include the board that the game is played upon as well as several different cards that belong to the game as well as the dice that are used within the game as well. The cards were created by texture mapping the card onto a thin BoxBufferGeometry object to give the illusion to the user that there is a card on the table. Creating the dice on the other hand was a more challenging process as it required me to learn how to map each face of the dice separately and join them together in order to make a cube that resembles the dice, I initially looked up some example code on how to do this [6] and while it worked, I felt like there was a better way of implementing the code needed to get the required effect. Instead of creating image objects, I decided that it would be more efficient to use one of the inbuilt library JavaScript files within three.js to also get the desired effect. In summary, I modified the code so that it used a TextureLoader class to have the same desired effect.



```
// Creates the dice that are on top of the table.
// Each face of the dice is an image that is stored within an array. The image is searched by the for loop seen below.
// The BoxGeometry object is then created and the array is used to insert the images onto each face of the object.
// Author Jake McDonagh
function drawDice() {
var dice = new THREE.Object3D();
var textureLoader = new THREE.TextureLoader();
var diceMaterials = [];

for (var i=0; i<6; i++) {
    var texture = textureLoader.load('images/dice' + i + '.png');
    var mat = new THREE.MeshBasicMaterial({color: 0xffffff, map: texture});
        diceMaterials.push(mat);
}

var diceGeo = new THREE.BoxGeometry(2, 2, 2, 1, 1, 1);
var diceMesh = new THREE.Mesh(diceGeo, new THREE.MeshFaceMaterial(diceMaterials));
    diceMesh.position.set(-50, 23, 0);
    dice.add(diceMesh);
    return dice;
}
```

Fig. 3. The code on how to map each face of a geometric object separately.

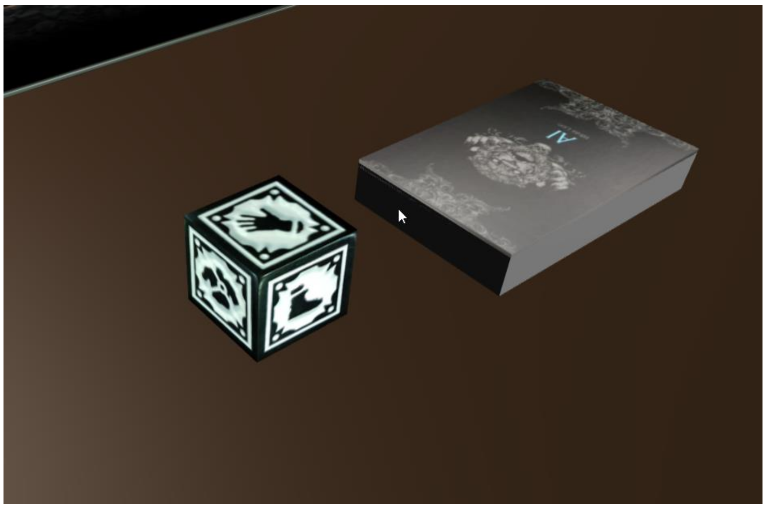


Fig. 4. The result of the drawDice function.

5 Hallway

This section of the scene uses the following: carpet2.png, waterfall.jpg, whitewall.jpg.

This room is the smallest room in the scene, only consisting of a simple corridor that utilizes a textured carpet floor [7] as well as the painted white texture for both the walls and ceiling [2]. The room also consists of two beige coloured lamps that are similar to the one found within the kitchen area as well as a similar picture frame of a waterfall that is hanged up on the wall [8]. The corridor also has two doors on each end that are created from BoxBuffer geometries with the handles being made with small ConeBuffer geometries which serve the purpose of showcasing how the light is capable of being blocked off as the light approaches either the kitchen or the bedroom. There is also a table object at the end of the corridor.

The hallway room is here to simply connect both the kitchen and bedroom areas of the house together in a realistic setting.

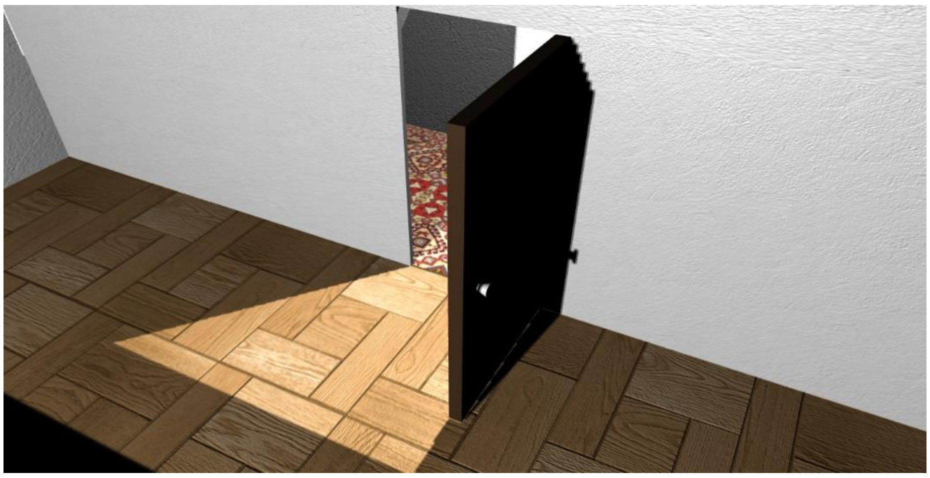


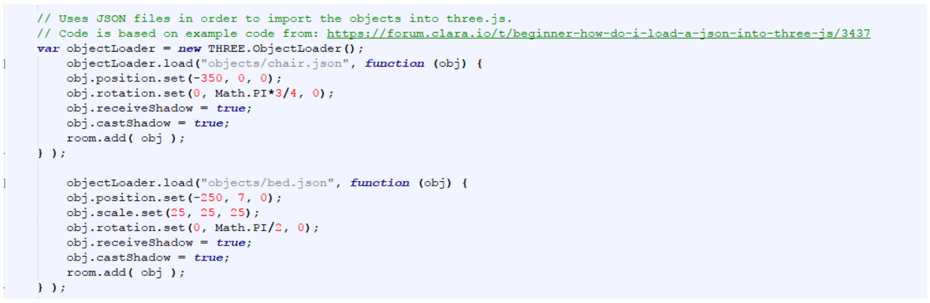
Fig. 5. The light from the hallway being blocked by the door by the kitchen entrance.

6 Bedroom

This section of the scene uses the following: bedroomfloor.jpg, ceiling.jpg, kingdomDeathBoxSice0.png, kingdomDeathBoxSice1.png, kingdomDeathBoxSice2.png, kingdomDeathBoxSice3.png, kingdomDeathBoxSice4.png, kingdomDeathBoxSice5.png, poster.jpg, wood.jpg, chair.json, bed.json

The Bedroom is most likely to be the last room that the user will see after traversing the hallway section of the scene. Like the kitchen, the bedroom also makes use of both texture mapping and shadow mapping to ensure that the room looks interesting. The floor consists of a carpet [9] in one corner of the room with the rest of the floor being the same wooden texture from previous rooms [1]. The walls also use the painted white texture that can be seen in the other rooms as well [2]. The ceiling uses a ceiling texture to ensure that the BoxBuffer geometry used has a ceiling-like quality [10]. Like the textures before it, all of the textures here have also been bump mapped to give the textures a photorealistic look to them.

Within this room, there is a bed [11] object and a chair [12] object, both of which are objects that have been imported from. json files that can be found within the 'objects' folder from the zip. In order to import these objects into three.js, I had to use an objectLoader class. After doing some research online, I eventually came across some example code which showed me how to import these json objects [13] which I then modified and repurposed so that it could work with the code that I had written thus far.



```
// Uses JSON files in order to import the objects into three.js.
// Code is based on example code from: https://forum.clara.io/t/beginner-how-do-i-load-a-json-into-three-js/3437
var objectLoader = new THREE.ObjectLoader();
    objectLoader.load("objects/chair.json", function (obj) {
    obj.position.set(-350, 0, 0);
    obj.rotation.set(0, Math.PI*3/4, 0);
    obj.receiveShadow = true;
    obj.castShadow = true;
    room.add( obj );
} );

    objectLoader.load("objects/bed.json", function (obj) {
    obj.position.set(-250, 7, 0);
    obj.scale.set(25, 25, 25);
    obj.rotation.set(0, Math.PI/2, 0);
    obj.receiveShadow = true;
    obj.castShadow = true;
    room.add( obj );
} );
```

Fig. 6. The code that has been modified to load .json files into three.js. Note that I added extra properties to ensure that the objects were capable of producing shadows.

The Bedroom also contains a table with the box of Kingdom Death: Monster on top of it, which was recreated into three.js by taking pictures of my own physical copy of the game (which can be seen in one of the pictures in the kitchen area) and implementing each face of the box onto it's corresponding mesh of the object. There is also a poster on the wall of the Bedroom as well.

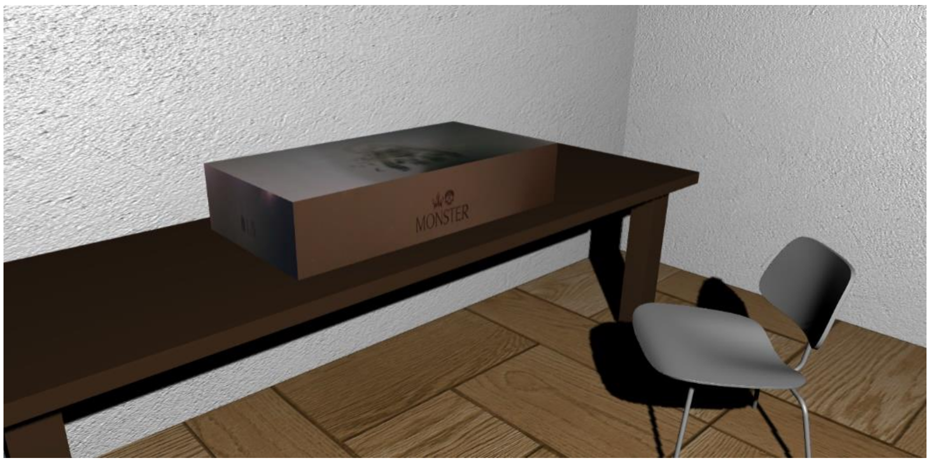


Fig. 6. The recreated box of Kingdom Death: Monster

7 Conclusion

Overall, I feel that the assignment has been a success for the most part. I was very happy with the outcome of the scene in terms of its visuals and lighting as I feel that I did an excellent job on using textures that worked together as well as ensuring that my objects had the appropriate shadow mapping, bump mapping and texture mapping to ensure that all of my meshes had reasonably realistic properties to them. I know that there are several improvements that I could make to my code however. The following improvements I could make to my scene are as follows:

- I was not able to make any proper implementations on any interactivity or animations. In the future I could add interactable light switches that allow the user to turn the lights on or off in a given room. I could also create a window and allow for snow animations to appear outside of it.
- I could add some background scenery and more windows to the house overall.
- I wasn't able to get a proper navigation controller to successfully work with the wasd or arrow keys. Further implementation to have the user navigate with mouse and keys instead of just the mouse would increase the quality of the work done.
- Add more props to the house such as bags, clothes, pens etc. to make the house feel more realistic.
- I could have also added all of my images into a single sprite sheet to allow for more efficient loading of the images used during this assignment.

References

1. WoodenTexture- https://i.pinimg.com/originals/89/58/02/895802f25575f7af5328805e750fc865.jpg
2. White wall Texture- https://farm9.staticflickr.com/8172/7961837172_68b5fc7060_b.jpg
3. Cat Picture- https://static.pexels.com/photos/177809/pexels-photo-177809.jpeg
4. https://static.pexels.com/photos/36717/amazing-animal-beautiful-beautifull.jpg
5. Kingdom Death: Monster reference- https://shop.kingdomdeath.com/products/kingdom-death-monster
6. How to map each face separately- https://stackoverflow.com/questions/13795354/verification-of-using-multiple-textures-with-three-js-cubes
7. CarpetHallwayTexture- http://www.publicdomainpictures.net/pictures/120000/velka/seamless-carpet-pattern.jpg
8. Waterfall Image - https://static.pexels.com/photos/6832/waterfall-beauty-lets-explore-lets-get-lost.jpg
9. CarpetBedroomTexture- http://www.luxurystandardhospitality.com/wp-content/uploads/2013/04/parquet2flooring3.jpg
10. CeilingTexture- http://4.bp.blogspot.com/-5xNMmxVjWKg/UA5bNp_0jaI/AAAAAAAAB4s/QXcwBUUPPKw/s1600/Seamless+wall+white+paint+stucco+plaster+texture.jpg
11. Bed Model - https://clara.io/view/8a80bc85-5eed-4d12-8f77-fd7824409cb2
12. Chair Model - https://clara.io/view/67bc637b-c528-44a0-bfbc-84335d12bcfa#
13. Importing JSON into three.js- https://forum.clara.io/t/beginner-how-do-i-load-a-json-into-three-js/3437